International Journal of Computational Geometry & Applications

Vol. 3. No. 3 (1993) 291-304

MAXIMUM INDEPENDENT SET OF A PERMUTATION GRAPH IN K TRACKS*

D. T. LEE and MAJID SARRAFZADEH
Department of Electrical Engineering
and Computer Science
Northwestern University
Evanston, IL 60208

Received 23 January 1992
Revised 3 May 1993
Communicated by W. L. Hsu

ABSTRACT

A maximum weighted independent set of a permutation graph is a maximum subset of noncrossing chords in a matching diagram (i.e., a set Φ of chords with end-points on two horizontal lines). The problem of finding, among all noncrossing subsets of Φ with density at most k, one with maximum size is considered, where the density of a subset is the maximum number of chords crossing a vertical line and k is a given parameter. A $\Theta(n \log n)$ time and $\Theta(n)$ space algorithm, for solving the problem with n chords, is proposed. As an application, we solve the problem of finding, among all proper subsets with density at most k of an interval graph, one with maximum number of intervals.

Keywords: Permutation graphs, maximum independent set, maximum chain, track assignment, plane sweep technique.

1. Introduction

Consider two rows of distinct integer points in the **xy**-plane. The *lower row* is at $\mathbf{y} = b_1$ and the *upper row* at $\mathbf{y} = b_2$, for some $b_1 < b_2$. A *chord* $N_i = (\ell_i, u_i)$ is a line segment passing through points (ℓ_i, b_1) and (u_i, b_2) of the plane where ℓ_i is the *lower terminal* and u_i is the *upper terminal* of N_i. A *matching diagram* consists of a collection $\Phi = \{N_1, \dots, N_n\}$ of chords. A subset of Φ is called an *independent set* if the corresponding chords are pairwise noncrossing. An independent set of maximum cardinality, among all independent sets, is called a *maximum independent set* (MIS). The density d_i^a of a subset Φ_a of Φ at a half-integer column $x = i + \frac{1}{2}$ (where i is an integer) is defined as follows.

$$d_i^a = \left|\{N = (\ell, u) \in \Phi_a \,|\, \ell < i + \tfrac{1}{2} < u \quad or \quad u < i + \tfrac{1}{2} < \ell\}\right|$$

The *density* d_a of a subset Φ_a of Φ is $\max_i(d_i^a)$, for all i.

*Supported in part by the National Science Foundation under grants CCR-8901815 and MIP-8921540.

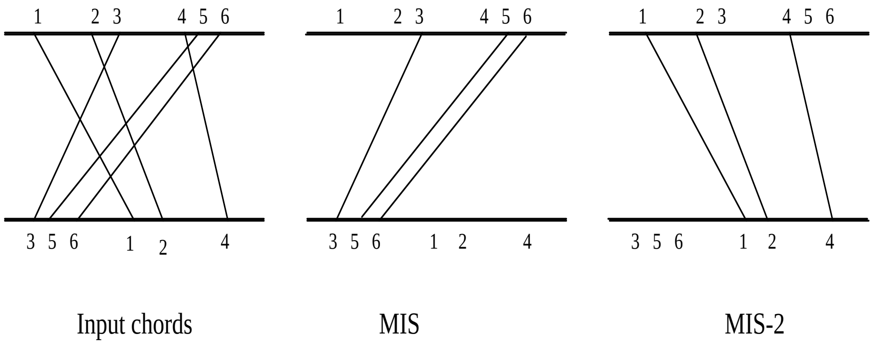


Figure 1: Density of maximum independent sets

Throughout this paper, we assume the input is a collection of (unsorted) pairs (ℓ_i, u_i). A geometric algorithm for obtaining a maximum independent set of Φ, running in $\Theta(n \log n)$ time, has been obtained, where $n = |\Phi|$.[1] However the density of the resulting MIS can be arbitrarily large (i.e., as large as n). MIS-k problem is to find, among all independent subsets of Φ with density at most k, one with maximum cardinality (see Figure 1). Note that MIS-n is the traditional MIS problem and MIS-1 problem is that of finding a maximum independent set of a collection of intervals – both problems can be solved in $\Theta(n \log n)$ time.[1,2] Weighted versions of MIS-n and MIS-1 problems can be solved also in $\Theta(n \log n)$ time. The converse problem – to find among all maximum independent sets in Φ one with minimum density – can be easily solved (by selecting at each "maxima level",[5] in a greedy fashion, a chord that minimizes density).

In Section 2, we study fundamental properties of MIS-k and discuss the main steps of the proposed algorithm. We present details of our implementation in Section 3. Finally, an application of the proposed algorithms in interval graphs is presented in Section 4.

2. Basic Concepts

Consider an instance $\Phi = \{N_1, \ldots, N_n\}$ of MIS-k problem. We want to find a noncrossing (planar) subset of Φ, with maximum cardinality, having density at most k. In the traditional MIS problem, where the goal is to find a maximum independent set irrespective of its density, only relative positions of the terminals are important. However in MIS-k, absolute positions of the terminals are important and cannot be arbitrarily changed. For example, in Figure 1, by changing absolute positions of some terminals, while maintaining their relative positions, we can increase the size of MIS for a fixed k. (Indeed, many fundamental properties employed in MIS-n problem do not hold for MIS-k, as we will describe.)

We will exploit geometric nature of MIS-k problem to design an efficient algorithm for it. To each net $N_i = (\ell_i, u_i)$ we associate a point $P_i = (x_i, y_i)$ in the **xy**-plane, where $x_i = \ell_i$ and $y_i = u_i$. In the point set $\Psi = \{P_1, \ldots, P_n\}$ we assume $x_i > 0$, $y_i > 0$ (otherwise we can shift the origin) and $x_i > x_j$ for $i > j$. Clearly, **x**- and **y**-coordinates of the points are distinct. Following[8] we say P_i *dominates*

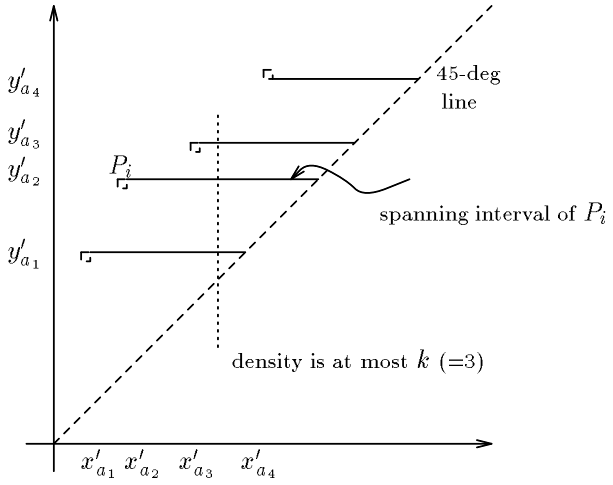


Figure 2: Non-containment and Density Property

P_j if $x_i > x_j$ and $y_i > y_j$. Points P_i and P_j are *crossing* of $x_i > x_j$ and $y_j > y_i$ or vice versa; otherwise, they are *noncrossing*. A subset of Ψ is called a *chain* if its points are pairwise noncrossing. A *maximum chain* (or simply, *maxchain*) is a chain with the maximum number of points among all chains. A *legal chain* is a chain with density (of the corresponding chords) at most k. A *legal maxchain* is a legal chain of maximum cardinality. Note that a maxchain of Ψ corresponds to a maximum independent set of Φ. Thus, to solve MIS-k, we need to obtain a chain of Ψ attaining certain properties – to account for its density. The transformation just described has been previously employed to solve various problems in a permutation graph (being the intersection graph of chords of Φ), for example, see Refs. [1,3,4,5,6,7].

With each point $P_i = (x_i, y_i)$ we associate an interval $I_i = (x'_i, y'_i)$, where $x'_i = \min(x_i, y_i)$, and $y'_i = \max(x_i, y_i)$. I_i is called the *spanning interval* of P_i. Consider a legal chain Ψ_{a_m} of Ψ, where $\Psi_{a_m} = \{P_{a_1}, \ldots, P_{a_m}\}$, with $x_{a_1} < \ldots < x_{a_m}$. We have the following properties.

Non-containment: No spanning interval of a point in Ψ_{a_m} contains that of the other, i.e., $x'_{a_1} < x'_{a_2} < \ldots < x'_{a_m}$ and $y'_{a_1} < y'_{a_2} < \ldots < y'_{a_m}$.

Density: For any value x, $x_{a_1} \leq x \leq x_{a_m}$ the number of spanning intervals containing x in its interior is at most k.

Non-containment property is to ensure the selected points are pairwise non-crossing and Density property bounds the density of the selected points by k. (See Figure 2). The following lemma is readily established.

Lemma 1 *A chain is legal if and only if it satisfies both the non-containment and the density property.*

We partition the set Ψ of points into two sets. P_i is called a *positive point* if $x_i \leq y_i$ and a *negative point* if $x_i > y_i$. For each chain $\Psi_{a_m} = \{P_{a_1}, \ldots, P_{a_m}\}, m \geq 1$, where $y'_{a_1} < y'_{a_2} < \cdots < y'_{a_m}$, we define a *header k-subchain* to consist of the last k points, $1 \leq k \leq m$, denoted $H(\Psi_{a_m})$. The point P_{a_m} is referred to as the *head* and the point $P_{a_{m-k+1}}$ the *tail* of the header subchain. If $k \geq m$, the header k-subchain is the chain Ψ_{a_m} itself, and P_{a_1} is the tail. Let $\mathcal{H}(\Psi_{a_m}) = (y'_{a_{m-k+1}}, y'_{a_{m-k+2}}, \ldots, y'_{a_m})$, i.e., $\mathcal{H}(\Psi_{a_m})$ consists of the list of right endpoints of the spanning intervals associated with the header k-subchain. As each chain has a unique header k-subchain which has a unique *head* and *tail*, we shall refer to the head and tail of the header k-subchain simply as the head and tail of the chain. We will also omit the parameter k if it is clearly understood, and the word *legal*, as we are only interested in legal chains.

Associated with the head P_i of the header subchain $H(\Psi_i)$ we define a *critical value*, denoted, $K(P_i)$, as follows. Let d_i denote the density of the spanning intervals of the points in $H(\Psi_i)$. If $d_i < k$, $K(P_i) = x'_i$, the left endpoint of spanning interval I_i; otherwise $(d_i = k)$ $K(P_i) = y'_j$, the right endpoint of spanning interval I_j of the tail P_j of Ψ_i. In the former case the chain is said to be *non-critical*, and in the latter it is said to be *critical*.

As each chain has a unique head, the critical value associated with the head is also referred to as the critical value associated with the chain. We say that the chain Ψ_i is *extendible* by point P_j, or P_j *matches* Ψ_i, if P_j dominates P_i, and $x'_j \geq K(P_i)$, i.e., P_j can be appended to chain Ψ_i to obtain a new chain $\Psi_j = \Psi_i \cup \{P_j\}$. P_j becomes the new head of Ψ_j and the new tail will be the next point in $H(\Psi_i)$ following the tail in the header subchain of Ψ_i, if $|H(\Psi_i)| = k$, and is the same as the old one, otherwise.

Let $\mathcal{R}$ denote a set of chains $\mathcal{R} = \{\Psi_{i_1}, \Psi_{i_2}, \ldots, \Psi_{i_t}\}$ extendible by point P_j. The chain Ψ_{i_q} and P_j are said to be a *canonical match*, if Ψ_{i_q} is such that $\mathcal{H}(\Psi_{i_q})$ is the smallest lexicographically. We say that the chain $\Psi_j = \Psi_{i_q} \cup \{P_j\}$ is obtained by *canonical* extension of Ψ_{i_q} to P_j. A chain Ψ_i is said to be a *canonical* chain, if it satisfies one of the following:

- Ψ_i contains only one point.
- If Ψ_i contains more than one point, i.e., $\Psi_i = \{P_{i_1}, P_{i_2}, \ldots, P_{i_t}\}$, $i_1 < i_2 < \ldots < i_t$, and $t > 1$, then each chain Ψ_{i_j} is obtained by canonical extension of $\Psi_{i_{j-1}}$ to P_{i_j}, where chain Ψ_{i_j} contains points $P_{i_1}, P_{i_2}, \ldots, P_{i_j}$ in that order, $j = 1, 2, \ldots, t$, i.e., P_{i_j} and chain $\Psi_{i_{j-1}}$ are a canonical match.

Among all chains with P_i as their head let Ψ_i^* be the one with maximum cardinality. We say P_i is at level $|\Psi_i^*|$, or, *level number* of P_i is $|\Psi_i^*|$. Thus, the set Ψ of points can be (uniquely) partitioned into a collection of points at levels 1 through s, where s is the size of a legal maxchain (see Figure 3).† We shall consider an iterative method that extends a chain one point at a time. We claim that when considering chains to be extended by a new point P_i, the canonical extension will produce a solution as good as any other strategy. This is known as the *greedy* method.

†In MIS-n and related problems,[1,6] the points belonging to the same level are pairwise crossing. This fact was exploited in designing algorithms for problems related to MIS-n. However, in MIS-k two points belonging to the same level may not cross each other.

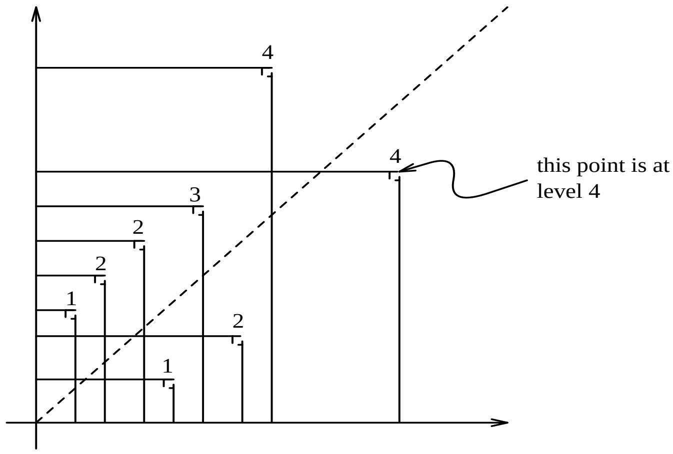


Figure 3: Level numbers when $k = 2$

Lemma 2 *Let S_i denote the set of points to the left of P_i and including P_i. Let Ψ_i denote a chain whose head $P_i \in S_i$ is at level ℓ. There always exists a canonical chain Ψ'_i of size ℓ containing P_i as head.*

Proof: Let Ψ_i contain points $P_{j_1}, P_{j_2}, \ldots, P_{j_\ell}, j_1 < j_2, < \ldots, < j_\ell$, and $P_{j_\ell} = P_i$. We prove that for each chain Ψ_t with head P_{j_t}, $t < l$, we have a corresponding canonical chain Ψ'_{j_t}, which together with $P_{j_{t+1}}$ forms a canonical match (l is the induction parameter).

Let $\Gamma_i \subseteq \Psi$ denote the set of points dominated by P_i. Consider the basis when $t = 1$. Let $P'_{j_1} \in \Gamma_{j_2}$ be the point with the smallest y'_{j_1}. Since P_{j_2} matches $\Psi_{j_1} = \{P_{j_1}\}$, $\Psi'_{j_1} = \{P'_{j_1}\}$ and P_{j_2} are a canonical match. Assume now that chain $\Psi'_{j_{l-2}} = \{P'_{j_1}, \ldots, P'_{j_{l-2}}\}$ and $P_{j_{l-1}}$ are a canonical match and $\Psi'_{j_{l-2}}$ is a canonical chain of size $l-2$. Let $P'_{j_{l-1}}$ be the point in Γ_{j_l} that matches $\Psi'_{j_{l-2}}$ such that $\mathcal{H}(\Psi'_{j_{l-1}})$ is lexicographically minimum, where $\Psi'_{j_{l-1}} = \Psi'_{j_{l-2}} \cup \{P'_{j_{l-1}}\}$. Since P_{j_l} matches $\Psi'_{j_{l-2}} \cup \{P_{j_{l-1}}\}$, it also matches $\Psi'_{j_{l-1}}$, and P_{j_l} and $\Psi'_{j_{l-1}}$ are a canonical match. We therefore have a canonical chain $\Psi'_{j_l} = \Psi'_{j_{l-1}} \cup \{P_{j_l}\}$ of size l with P_{j_l} as the head. $\square$

Note that in a non-critical chain, the density of the header subchain is always less than k, so the critical value of the head P_i is $x'_i = min(x_i, y_i)$. We thus have the following lemma.

Lemma 3 *Any point that dominates the head of a non-critical chain, matches it.*

3. Details of Implementation

In this section we present a detailed algorithm for solving MIS-k problem. We employ the plane-sweep technique,[8] that is, we process the points $\Psi = \{P_1, \ldots, P_n\}$ from left to right. When P_i is encountered, we assume that the level numbers of previously scanned points have been computed. We proceed to find, among all chains with head P_i, one with maximum cardinality, which gives the level number of P_i. As shown in Lemma 2, it suffices to find a canonical chain Ψ_j for some $j < i$

of maximum cardinality such that Ψ_j and P_i are a canonical match.

We assume further that for each previously scanned point P_j, we have also calculated its critical value $K(P_j)$ (of the canonical chain Ψ_j) and obtained a *linked list* representation of the header subchain $H(\Psi_j)$. Note that each header subchain $H(\Psi_j)$, consists of points arranged in *ascending* order of their y' values, i.e., the right endpoints of their spanning intervals. For ease of reference, the list is represented as $\hat{\mathcal{H}}(\Psi_j) = (y'_{j_1}, y'_{j_2}, \ldots, y'_{j_k})$, where y'_{j_1} corresponds to tail($\mathcal{H}(P_j)$) and y'_{j_k} corresponds to head($\mathcal{H}(P_j)$).

Now, consider point P_i and the set Γ_i of points dominated by P_i. In Γ_i we find a chain headed by point P_j such that $K(P_j) \leq x_i$, and its level number λ_j is maximum. If more than one chain satisfies the condition, select the one whose associated header list $\hat{\mathcal{H}}(P_j)$ is *lexicographically minimum* – canonical extension. The level number of P_i is $\lambda_i = \lambda_j + 1$ and we have a pointer from P_i to P_j. We obtain the canonical chain $\Psi_i = \Psi_j \cup \{P_i\}$, and $\hat{\mathcal{H}}(P_i)$ becomes $\hat{\mathcal{H}}(P_j)-$ tail($\mathcal{H}(P_j)$)$\cup\{y'_i\}$, i.e., the tail of $\mathcal{H}(P_j)$ is deleted and y'_i is appended. Set $K(P_i)$ to be $max(x'_i, Y)$, where $x'_i{=}min(x_i, y_i)$, and $Y=$ tail($\mathcal{H}(P_i)$). We distinguish two cases.

CASE 1) P_i is a positive point:

We store $\hat{\mathcal{H}}(P_i)$, and its level number λ_i, in a structure T at location $y'_i = max(x_i, y_i) = y_i$.

Suppose $\Psi_i = \Psi_j \cup \{P_i\}$. If $x_i < Y$, where $Y=$ tail($\mathcal{H}(P_i)$), then $K(P_i) = Y$. Note that in this case, the density of Ψ_i must be k and Ψ_i is critical. Thus the leftmost point that matches Ψ_i must have **x**-coordinate not smaller than Y. See Figure 4a. To ensure the above matching condition, we need to do the following. We temporarily set λ_i in T to zero and when the sweep-line reaches $\mathbf{x} = Y$ we will restore the actual value of λ_i. This is equivalent to saying in the **x**-interval $[x_i, Y)$ no point matches Ψ_i.

In this case P_i will be entered in another structure I at location $y_q(= Y)$, which corresponds to the tail of $\mathcal{H}(P_i) = P_q$, so that when we scan passing the value, y_q, i.e., $x_l \geq K(P_i) = y_q$ for some l, we need to restore the value λ_i in T. Otherwise, $(x_i \geq Y)$ Ψ_i is non-critical, the actual value λ_i is stored directly in T at y'_i. Therefore T should support *insertions, updates* and *Max* operations, i.e., given a value y_i, the point P_j with the maximum λ_j value and minimum $\hat{\mathcal{H}}(P_j)$ can be located efficiently.

CASE 2) P_i is a negative point:

This case is slightly different from the previous one in the treatment of λ values. We store $\hat{\mathcal{H}}(P_i)$ and its level number λ_i at location $K(P_i)$ in the structure T. See Figure 4b.

To see this, let P_q be the tail of $\mathcal{H}(P_i)$. If Ψ_i is critical, i.e., $y_i < x_q$, and $K(P_i) = x_q$, the bottommost point that matches Ψ_i must have **y**-coordinate not smaller than x_q. As the next point to be examined lies to the right of x_i and $x_i > x_q$, we can store λ_i in T at $K(P_i)$ directly, since $y_i < x_i$, and $y_q < x_q$.

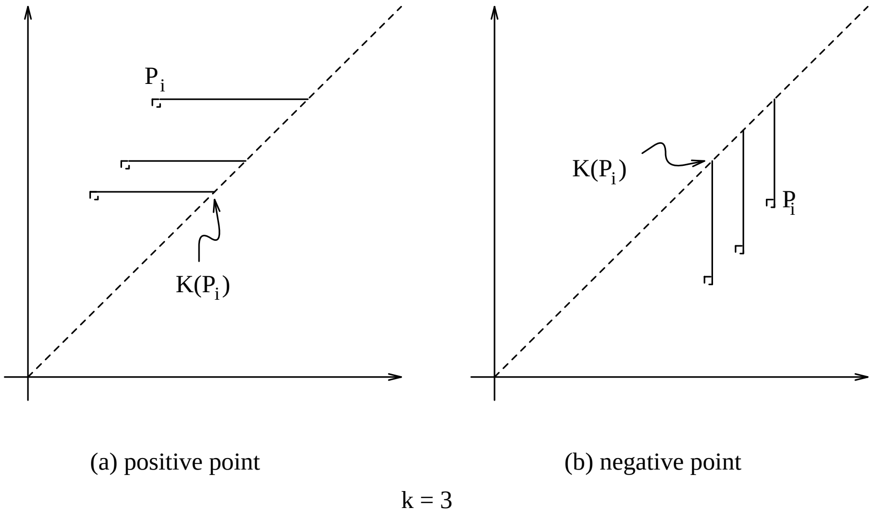


Figure 4: Cases 1 and 2

The structure T is a height-balanced tree with n leaves and is organized with respect to **y**-coordinates of the points. To be more precise, T is a *priority* queue with values $(\lambda_i, -\hat{\mathcal{H}}(\Psi_i))$ stored at the leaf corresponding to a point P_i at location Y, where $Y = y_i$ if P_i is a positive point, and $Y = K(P_i)$ otherwise. Values $(\lambda_v, -\hat{\mathcal{H}}_v)$ are stored at internal node v such that $(\lambda_v, -\hat{\mathcal{H}}_v)$ is lexicographically maximum among all such values in the leaves of the subtree rooted at v. In T we need to spend $O(k)$ time at each node to compare two header sub-chains. The next lemma shows that $\hat{\mathcal{H}}(\Psi_i)$ does not have to be stored explicitly.

Consider processing a point P_i. Let $\Psi_\pi = \{P_{\pi_1}, \ldots, P_{\pi_l}\}$ and $\Psi_\sigma = \{P_{\sigma_1}, \ldots, P_{\sigma_l}\}$ be two canonical maxchains that are extensible by P_i. First we observe that if $P_{\pi_l} = P_{\sigma_l}$ then $\hat{\mathcal{H}}(P_{\pi_l}) = \hat{\mathcal{H}}(P_{\sigma_l})$, for each canonical chain that matches P_{π_l} – by definition – has the lexicographically smallest header subchain.

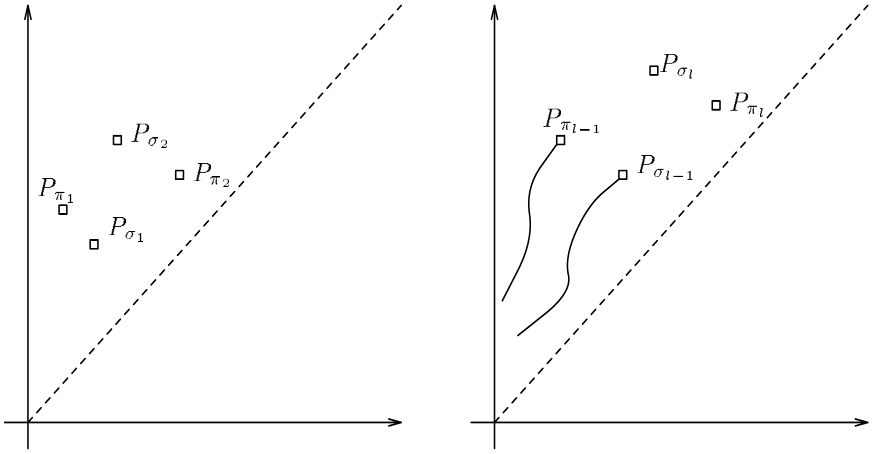
Lemma 4 *If $y'_{\pi_l} < y'_{\sigma_l}$ then $\hat{\mathcal{H}}(P_{\pi_l})$ is lexicographically smaller than $\hat{\mathcal{H}}(P_{\sigma_l})$.*

Proof: We prove this by induction on l.

Basis: Consider $l = 2$ (when $l = 1$, $y'_{\pi_l} = y'_{\sigma_l}$), if $y'_{\pi_1} < y'_{\sigma_1}$ then $\hat{\mathcal{H}}(P_{\pi_l})$ is lexicographically smaller than $\hat{\mathcal{H}}(P_{\sigma_l})$. Otherwise, if $y'_{\pi_1} > y'_{\sigma_1}$ (see Figure 5a), Ψ_π is not a canonical chain. Instead, $(P_{\sigma_1}, P_{\pi_2})$ is a canonical chain.

Inductive step: If $y'_{\pi_{l-1}} < y'_{\sigma_{l-1}}$, by the induction hypothesis and by the observation preceding the lemma, $\hat{\mathcal{H}}(P_{\pi_{l-1}})$ is lexicographically not larger than $\hat{\mathcal{H}}(P_{\sigma_{l-1}})$. Thus, $\hat{\mathcal{H}}(P_{\pi_l})$ is lexicographically smaller than $\hat{\mathcal{H}}(P_{\sigma_l})$. Otherwise, if $y'_{\pi_{l-1}} > y'_{\sigma_{l-1}}$ (Figure 5b), then Ψ_π is not a canonical chain – Indeed, $(P_{\sigma_1}, \ldots, P_{\sigma_{l-1}}, P_{\pi_l})$ is a canonical chain. □

Based on Lemma 4, T is a *priority* queue with values $(\lambda_i, -y'_i)$ stored at the leaf corresponding to a point P_i at location Y, where $Y = y_i$ if P_i is a positive point, and $Y = K(P_i)$ otherwise. Values $(\lambda_v, -y'_v)$ stored at internal node v are such that the pair, $(\lambda_v, -y'_v)$, is lexicographically maximum among all such pairs of values stored



(a) basis (b) inductive step

Figure 5: Proof of Lemma 4

in the leaves of the subtree rooted at v. By virtue of Lemma 4, a lexicographically maximum $(\lambda_v, -y'_v)$ implies a lexicographically maximum $(\lambda_v, -\hat{\mathcal{H}}(\Psi_v))$. Each leaf node points to the previous point in its header subchain, if its header subchain is non-empty.

When we process a positive point P_i, the value λ_i in $(\lambda_i, -y'_i)$ stored in the corresponding leaf node at y_i in T may be set to contain a temporary value Temp-i (being zero), which is made permanent and contains the actual value λ_i at a later time. To efficiently obtain the canonical chain with P_i as the head we also store a pointer to the head of the chain that P_i matches (i.e., P_j in the above discussion). I is a linear array that keeps track of those positive points whose λ values need to be reset to permanent. It is organized with respect to increasing order of **y**-coordinate; each entry y_j is a list of points P_i whose $K(P_i) = y_j$ and whose λ values need to be reset.

When we process a negative point P_i we store the pair $(\lambda_i, -y'_i)$ at location $K(P_i)$, and a pointer to the head of the chain that P_i matches (i.e., P_j in the above discussion).

We perform the following operations on the structures just described.

MAX(y_i) :	Find the maximum value $(\lambda_j, -y'_j)$ in $[1, y_i]$ of T and return the result $(max, index)$, where max contains λ_j, and $index$ contains j.
INS(y_i, λ_i, y'_i):	Insert (in Temp location) at y_i of T the value $(\lambda_i, -y'_i)$.
SET(y_i) :	Set the temporary value of y_i in T into its permanent value.
IN.I(y_j, P_i) :	Insert P$_i$ at location y_j of I.

OT.I(x_i) :	Return all points in the list located at position $y_i < x_i$ in I and have them removed from I.
PTR(P_i, P_j) :	Add a pointer from P$_i$ to P$_j$.
tail(P_i) :	Find $tail(\hat{\mathcal{H}}(P_i))$.

Next, we will describe in details how to update the structures. All values are initially set to zero. We assume points $P_1, \ldots, P_{i-1}$ have been processed and the structures have been accordingly updated.

```
Algorithm LEVEL (Ψ):
for each point P_i do
    {P_a1, ..., P_am} := OT.I(x_i)
    for 1 ≤ j ≤ m do SET(y_aj)
    y'_i := max(x_i, y_i)
    (max, index) := MAX(y_i)
    if P_i is a positive point
    then begin
            INS(y_i, max + 1, y'_i)
            j := index
            PTR(P_i, P_j)
            Y := tail(P_i)
            if x_i < Y then IN.I(Y, P_i) else SET(y_i)
        end
    else begin {P_i is a negative point}
            j := index
            PTR(P_i, P_j)
            Y := tail(P_i) :
            INS(Y, max + 1, y'_i)
            SET(Y)
        end
```

Lemma 5 *Algorithm LEVEL correctly computes the level number of all points.*

Proof: Assume level number of points $P_1, \ldots, P_{i-1}$ have been correctly computed and each point P_t is associated with a lexicographically minimum $\hat{\mathcal{H}}(P_t)$,

$t = 1, 2, \ldots, i-1$. Consider the set $\Gamma_i = \{P_{a_1}, \ldots, P_{a_m}\}$ of points dominated by P_i. Let λ_{max} denote the maximum level of points in Γ_i, that is, $\lambda_{max} = \max_{j=1}^{m}(\lambda_{a_j})$.

Let P_i match a chain Ψ_t of size λ_{long} with head P_t and P_i is a canonical match of Ψ_t. We have $\lambda_i = \lambda_{long} + 1$ where $\lambda_{long} \in \{\lambda_{max} - 1, \lambda_{max}\}$ for there exists at least one chain of size λ_{max} whose head is in Γ_i and we can replace the head with P_i.

CASE 1) $\lambda_i = \lambda_{max} + 1$

This is the maximum possible level number for P_i since we have assumed all points in Γ_i have the correct level number. Since P_i matches Ψ_t, i.e., $min(x_i, y_i) > K(P_t)$, $\lambda_t = \lambda_{max}$ must have been set permanent before we process P_i. Since MAX(y_i) returns the point P_t with minimum $\hat{\mathcal{H}}(\Psi_t)$, $\hat{\mathcal{H}}(\Psi_i)$ and λ_i will be computed and correctly entered into T.

CASE 2) $\lambda_i = \lambda_{max}$

In this case for all points P_j in Γ_i with level number λ_{max}, $K(P_j) > x_i'$, i.e., their λ values either remain temporary or are not involved when MAX(y_i) is called. Since for all points P_t with level number $\lambda_{max} - 1$, $K(P_t) < x_i'$, they must all be permanent and hence available when MAX(y_i) is called. $\hat{\mathcal{H}}(\Psi_i)$ and λ_i are computed and correctly entered into T.

□

Operations described in Algorithm LEVEL are straightforward except for the operation $tail(P_i)$. It requires a special treatment of the tree T. One approach is to follow the pointers, starting at the leaf corresponding to P_i, to obtain Y. This approach requires $O(\log n + k)$ time to process each point P_i. That is, the algorithm runs in $O(n(\log n + k))$ time and requires $O(n)$ space. Next we show how to improve the time complexity of this direct approach.

Consider a path $\mathcal{P}$ consisting of m vertices. The vertex with out-degree zero is at level one. A vertex with distance l from the vertex v_1 at level 1 is at level $l + 1$, $1 \leq l \leq m - 1$. The vertex at level l is denoted by v_l. Edges of $\mathcal{P}$ are

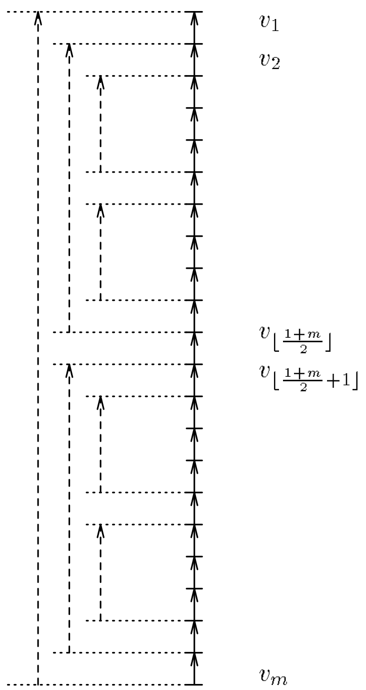


Figure 6: Dashed and solid edges in $\mathcal{P}^*$

called *solid edges.* We add a collection of *dashed edges* to $\mathcal{P}$ as follows. The set of vertices between levels 1 and m is called a *block* and is denoted by $B(1, m)$. Each block $B(a, b)$, $b - a > 3$, is partitioned into two subblocks $B(a + 1, \lfloor \frac{a+b}{2} \rfloor)$ and $B(\lfloor \frac{a+b}{2} + 1 \rfloor, b - 1)$, and there is a dashed edge of length $b - a$ between vertex v_a to vertex v_b of block $B(a, b)$. See Figure 6. The new graph, being the union of solid and dashed edges, is denoted by $\mathcal{P}^*$. Note that there is a path from any vertex v_i to any other vertex v_j, $i \geq j$, of length $O(\log m)$. In $\mathcal{P}^*$ each vertex has degree at most two: one contributed by a dashed edge and one by a solid edge. Next, we will use the just described technique for finding the tail of a point in Algorithm LEVEL.

Consider a *forest* with vertices corresponding to points $\{P_1, \ldots, P_n\}$ in which the total number of levels is m. Its edges correspond to pointers added in Algorithm LEVEL. That is, when $PTR(P_i, P_j)$ is executed, an edge from P_i to P_j is introduced. By introducing a point at $(-\infty, -\infty)$, the given forest becomes a tree (certainly, that point will not be considered when calculating the levels). We denote this tree by T_{ptr}. Edges of the tree are the *solid edges*. When we process P_i, we add a *dashed edge* from P_i at level l_1 to the point in the chain containing P_i, at level l_2, $l_1 > l_2$, if there is a dashed edge between v_{l_1} to v_{l_2} in $\mathcal{P}^*$. Thus, dashed edges will be added dynamically. To add this dashed edge, we need to follow exactly two other dashed edges and three solid edges, if $l_1 - l_2 > 3$. In case $l_1 - l_2 = 3$, we need to follow exactly three solid edges (see Figure 6). Thus, it takes $O(1)$ to add a dashed edge. The new tree, containing solid and dashed edges is denoted by T^*_{ptr}. We can find the tail of a point P_i in T^*_{ptr} in $O(\log n)$ time. Note that for our purpose, dashes edges of length greater than k need not be added to T^*_{ptr}. Thus, it takes $O(\log k)$ time to find the tail of a point. Processing P_i requires $O(\log n + \log k)$ time.

Theorem 1 *Algorithm LEVEL correctly computes the level number of all points in* Ψ *in* $\Theta(n \log n)$ *time and* $\Theta(n)$ *space, where* $n = |\Psi|$.

Proof: As proved in Lemma 4, the level number of each point is correctly computed. Each operation on T and T^*_{ptr} takes $O(\log n)$ time per point. Insertion and deletion of each point in I requires $O(1)$ time. Thus, the entire algorithm runs in $O(n \log n)$ time. The structure T and T^*_{ptr} take O(n) space, and I requires $O(n)$ space.

Note that MIS-n problem and MIS-1 problem require $\Omega(n \log n)$ time to find an optimal solution.[1,2] Thus, MIS-k, for an arbitrary k, also requires $\Omega(n \log n)$ time. Therefore the proposed algorithm is both time and space optimal. $\Box$.

4. Discussion and Conclusion

Here, we consider an application of MIS-k (and MWIS-k). Consider a set of intervals $\mathcal{I} = \{I_1, \ldots I_n\}$, where $I_i = (\ell_i, r_i)$. A *proper* subset of $\mathcal{I}$ is a subset where no interval contains any other intervals. That is, there are no two intervals I_i and I_j with $\ell_i < \ell_j < r_j < r_i$. MPS-$k$ problem is to find, among all proper subsets of $\mathcal{I}$ with density at most k, one with maximum cardinality. We can transform MPS-k problem into MIS-k problem. To each interval $I_i = (\ell_i, r_i)$ we associate a chord $N_i = (\ell_i, r_i)$ (see Figure 7). Let $\mathcal{N}$ denote $\{N_1, \ldots N_n\}$. Certainly a subset of chords in $\mathcal{N}$ are independent if and only if the corresponding subset of intervals

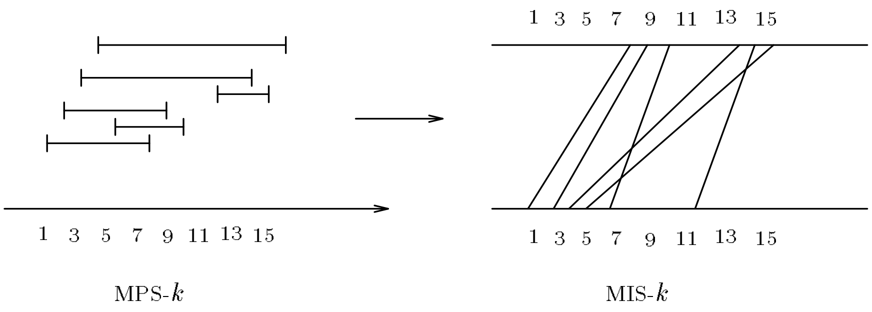


Figure 7: Transforming MPS-k into MIS-k

in $\mathcal{I}$ are proper. We conclude:

Theorem 2 *An arbitrary instance $\mathcal{I}$ of MPS-k can be solved in $\Theta(n \log n)$ time and $\Theta(n)$ space, where $n = |\mathcal{I}|$.*

A related problem of finding a maximum (unweighted) subset with density at most k of a collection of intervals was recently solved in $O(nlogk)$ time.[9] Intervals in the subset are arbitrary, that is, they are not pairwise proper. It is assumed that intervals are pre-sorted; otherwise, an additional $O(nlogn)$ time is required for pre-processing.

The complexity of the weighted version of MIS-k remains to be seen. We are also investigating the problem of finding among all two-chains (see Ref. 5 for a definition) with density k, one with maximum cardinatlity.

References

1. Atallah, M.J., Kosaraju, S.R., "An Efficient Algorithm for Maxdominance, with Applications," *Algorithmica*, Vol. 4, No. 2, pp. 221-236, 1989.
2. Gupta, U., Lee, D.T., Leung, J., "An Optimal Solution for the Channel Assignment Problem," *IEEE Transactions on Computers,* November 1979, pp. 807-810.
3. Liao, K.F., Lee, D.T., and Sarrafzadeh, M., "Planar Subset of Multi-terminal Nets," *INTEGRATION: The VLSI Journal,* No. 1, Vol. 10, Sep. 1990, pp. 19-37.
4. Lou, R.D. and Sarrafzadeh, M., " Circular Permutation Graph Family," *International Workshop on Discrete Algorithms and Complexity,* Fukuoka, Japan, November 1989, pp. 107-114; also to appear in *Discrete Applied Mathematics.*
5. Lou, R.D., Sarrafzadeh, M., and Lee, D.T., "An Optimal Algorithm for the Maximum Two-chain Problem," *Proceedings of the First SIAM-ACM Conference on Discrete Algorithms,* San Francisco, CA, January 1990, pp. 149-15; also to appear in *SIAM Journal on Discrete Mathematics.*
6. Sarrafzadeh, M. and Lee, D.T., "A New Approach to Topological Via Minimization"

IEEE Transactions on Computer-Aided Design, Vol. 8, No. 8, August 1989, pp. 890-900.

7. Sarrafzadeh, M. and Lee, D.T., "Topological Via Minimization Revisited," manuscript, 1989; to appear in *IEEE Transactions on Computers.*
8. Preparata, F. P. and M. I. Shamos, *Computational Geometry: an Introduction*, Springer-Verlag, 1986.
9. M. Yannakakis anf F. Gavril, "The Maximum K-colorable Subgraph Problem for Chordal Graphs" *Information Processing Letters*, Vol. 24, 1987, pp. 133-137.